

U.S. DEPARTMENT OF COMMERCE  
PATENT AND TRADEMARK OFFICE

PATENT APPLICATION TRANSMITTAL LETTER

Att'y Dkt No.:

2207/315

Assistant Commissioner for Patents  
Washington D.C. 20231

Transmitted herewith for filing is the patent application of

Inventors: Chinna Prudvi and Derek Bachand

For: **TRANSACTION MANAGER AND CACHE FOR PROCESSING AGENT**

Enclosed are:

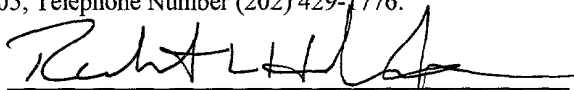
1. 15 sheets of specification, 4 sheets of claims, 1 sheet of abstract; and 3 sheets of formal drawing;

The filing fee has been calculated as shown below:

	NUMBER FILED	NUMBER EXTRA*	RATE (\$)	FEE (\$)
BASIC FEE			790.00	760.00
TOTAL CLAIMS	21 - 20 =	1	18.00	18.00
INDEPENDENT CLAIMS	4 - 3 =	1	78.00	78.00
MULTIPLE DEPENDENT CLAIMS PRESENT				0.00
FEE FOR RECORDATION OF ASSIGNMENT			40.00	0.00
*Number extra must be zero or larger			TOTAL	\$856.00

If applicant is a small entity under 37 CFR §§ 1.9 and 1.27, then divide total fee by 2, and enter amount here

2. The Office is authorized to charge the filing fee of **\$856.00** to Deposit Account No. **11-0600**. A duplicate copy of this paper is enclosed for that purpose.
3. Please direct all correspondence to Robert L. Hails, Jr., Kenyon & Kenyon, 1025 Connecticut Avenue, N.W., Washington, D.C. 20036-5405, Telephone Number (202) 429-1776.



Dated: December 16, 1998

Robert L. Hails, Jr. (Registration No. 39,702)

KENYON & KENYON  
1025 Connecticut Avenue, NW, Suite 600  
Washington, DC 20036  
Phone: (202) 429-1776  
Fax: (202) 429-0796

## TRANSACTION MANAGER AND CACHE FOR PROCESSING AGENT

### BACKGROUND

The present invention relates to an improved cache and transaction queue system in a processing agent.

Modern computer systems may include multiple processing agents that communicate with one another over an external bus. An "agent" may include a general purpose processor, a digital signal processor an input/output or memory chipset, a bridge interface to other buses in the system or other integrated circuit that communicates over the external bus.

Typically, agents exchange data through bus transactions. An external bus protocol defines signals to be used by the agents to implement the bus transactions. For example, an external bus protocol for the known Pentium® Pro processor, commercially available from Intel Corporation, defines a pipelined bus protocol in which a transaction progresses through as many as six phases. The phases include: an Arbitration phase, a Request phase, an Error phase, a Snoop phase, a Response phase and a Data phase. Data may be transferred between agents in the Data phase. According to the Pentium® Pro bus protocol,

up to 32 bytes of data may be transferred in a single bus transaction. Accordingly, an external memory in a computer system built around the Pentium® Pro bus protocol typically is organized into "data lines" having a 32 byte length. Other systems may operate according to other bus protocols and thereby define data lines of other lengths.

5 Agents typically include internal caches for storage of data. The internal cache operates at a higher clock rate than the external bus and, therefore, provides faster access to data than external memory. Known internal caches are populated by cache entries having the same length as the data lines of external memory. Thus, an internal cache in the Pentium® Pro processor possesses cache entries having 32 byte lengths. Again, cache entries  
10 of other systems may have different cache line lengths than the Pentium® Pro processor to match different data line lengths of their respective systems. However, in all known systems, the length of cache lines are the same as the length of the data lines.

Internal caches store not only data from external memory but also store administrative data related to the data from external memory. For example, the caches associate data with  
15 their external addresses. They may also store state information related to cache coherency functions. Storing such administrative data in the internal cache is disadvantageous because it increases the area of the internal cache when the agent is manufactured as an integrated circuit. The increased size of the internal cache translates into increased cost of the agent and increased power consumption of the internal cache.

20 Accordingly, there is a need in the art for an agent that possesses an internal cache with minimal area. There is a need in the art for such an agent that reduces the amount of administrative data stored in association with data from external memory.

## **SUMMARY**

Embodiments of the present invention provide a processing agent for use in a system  
25 that transfers data of a predetermined data line length in external transactions. The agent

may include an internal cache having a plurality of cache entries. Each cache entry may store multiple data line lengths of data.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

FIG. 1 is a block diagram illustrating a bus sequencing unit of an agent constructed in accordance with an embodiment of the present invention.

FIG. 2 is a block diagram illustrating an internal cache constructed in accordance with an embodiment of the present invention.

FIG. 3 is a block diagram of a queue entry of an external transaction queue constructed in accordance with an embodiment of the present invention.

FIG. 4 is a block diagram of a known multiple-agent processing system.

FIG. 5 is a block diagram of fields that may be present in a memory address according to an embodiment of the present invention.

FIG. 6 is a flow diagram of a method of an embodiment of the present invention.

## **DETAILED DESCRIPTION**

The present invention, in an embodiment, provides an internal cache in an agent having cache entries whose lengths are a multiple of the length of a data line. One address is stored for the multiple data lines thereby decreasing the area of the cache when the agent is manufactured as an integrated circuit. This is an improvement over traditional internal caches where address information is stored individually for each stored data line. The internal cache may be associated with an improved transaction queue system in which address information similarly is conserved.

In an embodiment, the principles of the present invention may be applied in a bus sequencing unit 200 ("BSU") of an agent, shown in FIG. 1. The BSU 200 includes an arbiter

210, an internal cache 220, an internal transaction queue 230 and an external transaction queue 240. The BSU 200 fulfills data requests issued by, for example, an agent core 100. An external bus controller 300 interfaces the BSU 200 to the external bus 400.

5 The arbiter 210 receives data request signals from not only the core 100 but also from a variety of other sources (not shown). Of the possibly several data requests received simultaneously by the arbiter 210, the arbiter 210 selects one of them and outputs it to the remainder of the BSU 200.

10 The internal cache 220 stores data in several cache entries (not shown in FIG. 1). It possesses control logic (also not shown) responsive to a data request to determine whether the internal cache 220 stores a valid copy of requested data. If so, it implements the request. For example, it may read or write data to the cache 220 as determined by a request type signal included in the data request signal.

15 The internal transaction queue 230 receives and stores data request signals issued by the arbiter 210. It coordinates with the internal cache 220 to determine if the requested data "hits" (was implemented by) the internal cache 220. If a data request "misses" the internal cache 220, the internal transaction queue 230 forwards the data request to the external transaction queue 240.

20 The external transaction queue 240 interprets data requests and generates external bus transactions to fulfill them. The external transaction queue 240 is populated by several queue entries. The external transaction queue 240 manages the agent's external bus transactions as they progress on the external bus 400. For example, when data is available in response to a read transaction, the external transaction queue 240 retrieves the data and forwards it to, for example, the core 100.

25 In an embodiment, the internal and external transaction queues 230, 240 may be replaced by a single transaction queue (not shown). In this embodiment, new requests are

loaded into the transaction queue. If the request hits the cache 220 the requests are removed from the queue.

The external bus controller 300 drives signals on the external bus 400 as commanded by the external transaction queue 240. During a single bus transaction, a predetermined length of data may be read to/from the agent via the external bus 400.

FIG. 2 illustrates a cache 500 constructed in accordance with an embodiment of the present invention. The cache 500 is appropriate for use as an internal cache 220 (FIG. 1). The cache 500 is populated by a number of cache entries 510. Each cache entry 510 includes a tag portion 520 and multiple data portions 530, 540 for storing copies of data from external memory (not shown). The data portions 530, 540 each store a quantity of data corresponding to a data line. The tag portion 520 stores address information identifying the data stored in the data portions 530 and 540. Cache entries 510 also store other administrative data in association with each data portion 530, 540 such as state information (shown as "S") and error correction codes (not shown). The cache 500 also includes a controller 550 that determines hits and misses as described below.

Embodiments of the present invention sever the relationships between "data line lengths" and "cache line lengths" that exist in agents of the prior art. Typically, in known agents, cache line length are the same as data line lengths. Embodiments of the present invention, by contrast, possess cache line lengths that are multiple data line lengths. Data from a single bus transaction would only partially fill a cache entry 510 of the internal cache 500.

Although each cache entry 510 stores multiple data lines, it includes only a single tag portion 520. The tag portion 520 identifies the address of the data stored in the data portions 530, 540. Data in adjacent data portions 530, 540 of a single cache line 510 are retrieved from adjacent locations in external memory (not shown). Thus, the number of tags 520 included in the cache 500 is reduced over traditional caches. The internal cache

500 may be comparatively smaller than known caches when manufactured as an integrated circuit.

The cache 500 may be an associative cache or a set associative cache.

FIG. 3 illustrates an entry 241 of an external queue 240 constructed according to an embodiment of the present invention. The queue entry 241 includes a primary entry 242 and a secondary entry 243. The primary entry 242 stores data related to a first bus transaction. It may include the address of the transaction, stored in an address field 244, and status information for the transaction, stored in a status field 245. Status information includes information regarding, for example, the request type, and the stage of the transaction (i.e. whether the transaction has been posted on the external bus 400 and the phase of the transaction). It may include data to be written externally pursuant to a write transaction. The status information also may indicate whether the first transaction is part of a multiple transaction sequence.

The secondary entry 243 stores status information related to a second bus transaction. In an embodiment, the secondary entry 243 includes only a status field 246 for the second transaction. The status field 246, like field 245, may store information regarding, for example, the request type and the stage of the transaction. The queue entry 241 may include as many secondary entries 243 as are necessary for the total number of entries (the one primary entry 242 and multiple secondary entries 243) to equal the number of data portions 530, 540 in the internal cache 220. In an embodiment, the primary-secondary structure of queue entries 241 may be repeated for every queue entry in the external transaction queue 240.

Using the primary-secondary queue entry structure of FIG. 3, the external transaction queue 240 either may post multiple transactions to fill an entire cache entry 510 (FIG. 2) or may post a single transaction to obtain a single data portion 530 or 540. A request cycle of the internal transaction queue 240 cycles through queue entries 241. When the request

cycle reaches a queue entry 241, control logic (not shown) examines the status field 245 of the primary entry 242, interprets the request and posts a transaction therefor. When the status field 245 indicates that the request is part of a multiple transaction sequence, the external transaction queue 240 interprets status information in status field 243, increments  
5 the address stored in field 242 to address a next data line and posts a second transaction therefor.

Optionally, a request type may be omitted from field 246 in the secondary entry 243. The request type typically is identical for all transactions stored in a single queue entry 241.

If, after a transaction is posted for the primary entry, the status field 245 indicates that the request is not part of the multiple transaction sequence, the request cycle advances to another entry 241 of the external transaction queue 240.  
10

FIG. 4 illustrates a multiple agent system constructed in accordance with an embodiment of the present invention. The agents 10-50 communicate with one another over the external bus 400. One of the agents 50 typically is a memory. The remaining agents 10-40 may share copies of the same data.  
15

Traditionally, in multiple agent systems, cache coherency rules are established to ensure that when an agent uses data, it uses the most current copy of the data that is present in the system. For example, the Pentium® Pro processor operates according to the MESI cache coherency scheme in which copies of data stored in an agent 10-40 are assigned one  
20 of four cache coherency states:

- **Invalid** state indicates that a copy of data is not available to the agent,
- **Shared** state indicates that the copy of data possesses the same value as is held in external memory; copies of data in shared state may also be stored by other agents.
- **Exclusive** state indicates that the agent is the only agent in the system (except a memory agent) that possesses a valid copy of the requested data.  
25



- **Modified state** indicates that the agent is the only agent in the system (except a memory agent) that possesses a valid copy of the requested data and the agent possesses a copy that is more current than the copy stored in external memory.

5 An agent determines what it may do with a copy of data based upon the state. For example, an agent cannot modify data in invalid or shared state without first posting an external bus transaction to acquire exclusive ownership of the data. Other processing systems may behave according to other cache coherency states. In the cache 500 of FIG. 2, state information may be stored in association with each data portion 530, 540 of a cache line  
10 (shown as "S").

15 Data states may change on a data line basis. Consider, for example, an example where an entire cache line 510 is stored with data in shared state. According to the MESI protocol, an agent 10 that stores data in shared state may read the data but may not modify the data without first obtaining ownership through an external bus transaction. Thereafter, another agent 20 may post an external bus transaction to obtain ownership of a data line stored in the cache entry 510 (stored in data portion 540). By protocol, the agent 10 marks its copy of the data as invalid. To implement this step, the agent changes the state of the data portion 540 to indicate that the data is invalid. Valid data remains in the other data portions 530 of the cache entry 510. Thus, although an agent 10 may fill cache entries 510 entirely with data, each data portion 530, 540 of the cache entry 510 need not necessarily  
20 change state in unison.

25 As noted with respect to FIG. 1, an internal cache 220 includes a controller 550 to determine whether a data request hits the cache. The cache 500 of FIG. 2 identifies two types of "hits:" a "cache" hit and a "tag" hit. A cache hit indicates that the cache 500 stores the requested data in cache coherency state that is valid for the request type of the data request. When a cache hit occurs, the controller 550 causes the data request to be executed on the corresponding data portion of the cache entry 510. A tag hit indicates that the

address of the new data request matches a tag stored in one of the cache entries 510, but that the cache entry does not store the requested data in a valid cache coherency state.

According to an embodiment of the invention, an external memory address may be populated by fields, shown in FIG. 5. The fields may include a tag field 710, an entry field 720 and an offset field 730. The tag field 710 may be used to determine whether a data request causes a cache hit, a tag hit or misses the cache 500 of FIG. 2.

When a data request is loaded into the cache 500, the controller 550 retrieves the tag field 710 from an address included in the data request. The controller 550 determines whether the tag field 710 matches data stored in any of the tag portions 520 of the cache entries 510. In an embodiment, the tag portions 520 are provided with match detection logic (not shown). The controller 550 forwards the tag field 710 to the match detection logic and detects a match signal therefrom. A tag match occurs when the tag field 710 matches data stored in one of the tag portions 720.

The entry field 720 identifies a specific area of the data portions 530, 540 of a matching cache entry. When a tag match occurs, the controller 550 reads the state information from the selected data portions (say 540). Based upon the request type of data request, the controller 550 determines whether the state of the data is valid for the data request. If so, a cache hit occurs.

In an embodiment, the BSU 200 operates according to the method of FIG. 6. A data request is posted to the BSU 200 (step 1010). The internal cache 220 determines whether the request hit the cache 220 (step 1020). If the request generates a cache hit, the internal cache implements the data request (step 1030). If the request generates a tag hit only, the external transaction queue 240 retrieves a data line (step 1040). If the request generates a cache miss and tag miss, the external transaction queue 240 retrieves a cache line (step 1050).

Accordingly, the present invention provides an internal cache and a transaction queue system for an agent having reduced area over known agents.

Several embodiments of the present invention are specifically illustrated and described herein. However, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention.

100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

**WE CLAIM:**

1. A processing agent adapted to transfer data of a predetermined data line length in an external transaction, the agent comprising an internal cache having a plurality of cache entries, each entry adapted to store multiple data line lengths of data.

2. The processing agent of claim 1, wherein the cache entries include a tag portion adapted to store address information.

3. The processing agent of claim 2, wherein the internal cache further comprises match detection logic for the tag portions, and control logic provided in communication with the match detection logic.

4. The processing agent of claim 1, wherein the cache entries include a cache coherency state field in association with each data line length of data.

5. The agent of claim 1, further comprising a transaction queue having a plurality of queue entries, the queue entries including a primary entry adapted to store address information and status information of a first external transaction and a secondary entry adapted to store status information of a second external transaction.

6. The agent of claim 4, wherein the status information of the first external transaction includes a field representing whether the first external transaction is part of a multiple transaction sequence.

7. The agent of claim 4, wherein the total number of primary and secondary entries equals the multiple number of data line lengths provided in the cache entries.

8. A processing agent, comprising a transaction queue having a plurality of a queue entries, the queue entries further comprising:

a primary entry including an address portion and status portion, the status portion provided for a first external transaction of the agent, and

5 a secondary entry including a status portion provided for a second external transaction.

9. The transaction queue of claim 8, wherein the status portion of the primary entry includes a field representing whether the first transaction is part of a multiple transaction sequence.

10. The transaction queue of claim 8, further comprising control logic adapted to cycle through the queue entries and post transactions therefrom.

11. A processing agent, comprising:

an internal cache having cache entries each adapted to store multiple data lines, and  
a transaction queue system adapted to post external transactions, each external  
15 transaction related to a single data line,

wherein the internal cache and the transaction queue system each receive data requests on a common input.

12. The processing agent of claim 11, wherein the internal cache and the transaction queue system communicate by signal lines.

20 13. The processing agent of claim 12, wherein the signals lines include a cache hit signal line and a tag hit signal line.

14. The processing agent of claim 11, wherein the transaction queue system comprises a plurality of queue entries, each queue entry comprising:

a primary entry including an address portion and status portion, the status portion provided for a first external transaction of the agent, and

5 a secondary entry including a status portion provided for a second external transaction.

15. The transaction queue of claim 14, wherein the status portion of the primary entry includes a field representing whether the first transaction is part of a multiple transaction sequence.

16. The transaction queue of claim 14, further comprising control logic adapted to cycle through the queue entries and post transactions therefrom.

17. A method of processing a data request within a processing agent comprising:  
posting the data request internally within the agent,  
determining whether the request hit the cache,  
15 when the request misses the cache, posting a sequence of external transactions to fill a cache line with data associated with the data request.

18. The method of claim 17, wherein the determining step includes:  
comparing address information of the data request with tags stored in the internal cache, and

20 identifying a cache miss when the address information does not match any stored tag.

19. The method of claim 18, wherein the determining step further includes:  
when address information matches a stored tag, reading cache coherency state information associated with the requested data, and

identifying a cache miss when the cache coherency state information is invalid for a request type of the data request.

20. The method of claim 17, further comprising, when the request hits the cache:  
determining whether the request hits a tag stored in the cache, and

5 if so, generating a single external transaction to read the requested data into the agent..

21. The method of claim 20, wherein the second determining step includes:  
comparing address information of the data request with tags stored in the internal  
cache, and

10 identifying a tag hit when the address information matches a stored tag.

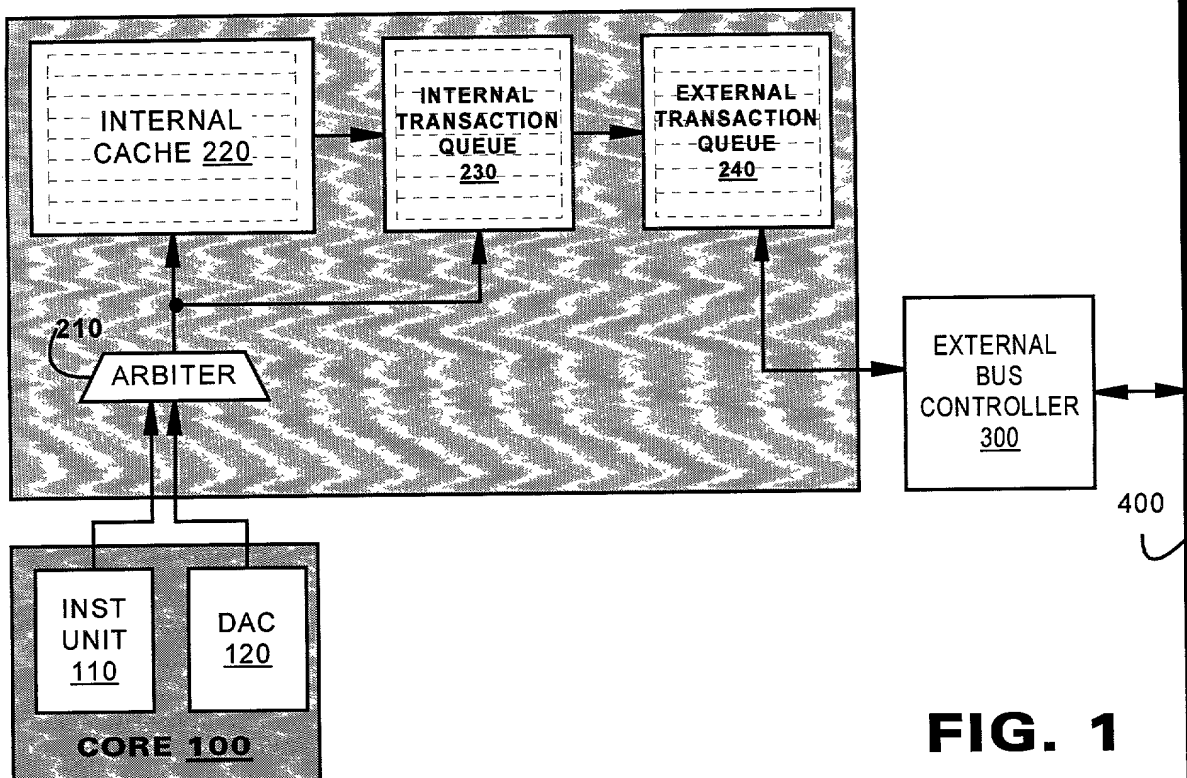
## ABSTRACT

A processing agent is used in a system that transfers data of a predetermined data line length during external transactions. The agent may include an internal cache having a plurality of cache entries. Each cache entry may store multiple data line lengths of data.

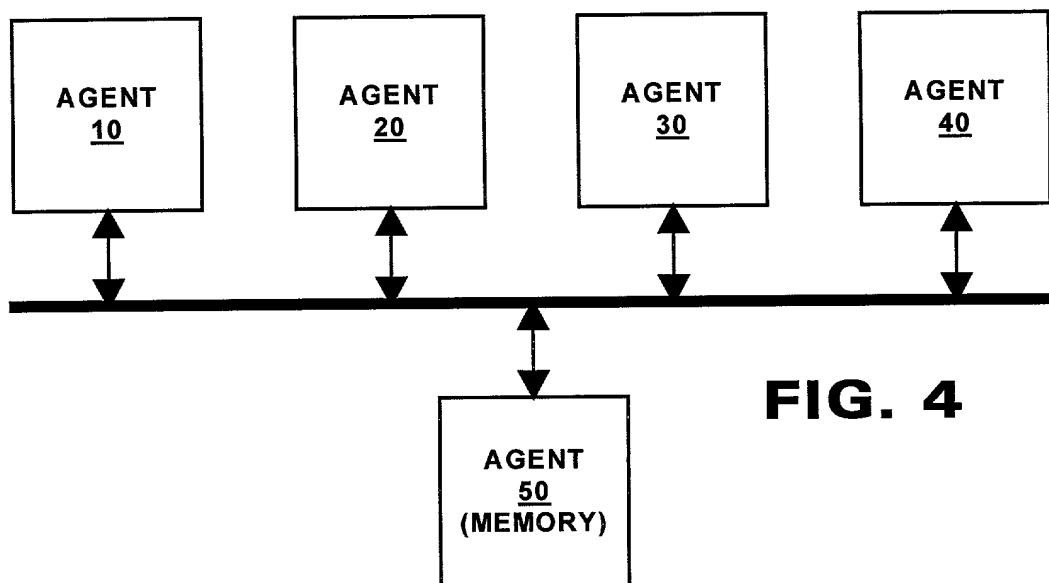
- 5 The agent further may include a transaction queue system having queue entries that include a primary entry including an address portion and status portion, the status portion provided for a first external transaction of the agent, and a secondary entry including a status portion provided for a second external transaction.

10  
15  
20  
25  
30  
35  
40  
45  
50  
55  
60  
65  
70  
75  
80  
85  
90  
95  
100  
105  
110  
115  
120  
125  
130  
135  
140  
145  
150  
155  
160  
165  
170  
175  
180  
185  
190  
195  
200  
205  
210  
215  
220  
225  
230  
235  
240  
245  
250  
255  
260  
265  
270  
275  
280  
285  
290  
295  
300  
305  
310  
315  
320  
325  
330  
335  
340  
345  
350  
355  
360  
365  
370  
375  
380  
385  
390  
395  
400  
405  
410  
415  
420  
425  
430  
435  
440  
445  
450  
455  
460  
465  
470  
475  
480  
485  
490  
495  
500  
505  
510  
515  
520  
525  
530  
535  
540  
545  
550  
555  
560  
565  
570  
575  
580  
585  
590  
595  
600  
605  
610  
615  
620  
625  
630  
635  
640  
645  
650  
655  
660  
665  
670  
675  
680  
685  
690  
695  
700  
705  
710  
715  
720  
725  
730  
735  
740  
745  
750  
755  
760  
765  
770  
775  
780  
785  
790  
795  
800  
805  
810  
815  
820  
825  
830  
835  
840  
845  
850  
855  
860  
865  
870  
875  
880  
885  
890  
895  
900  
905  
910  
915  
920  
925  
930  
935  
940  
945  
950  
955  
960  
965  
970  
975  
980  
985  
990  
995

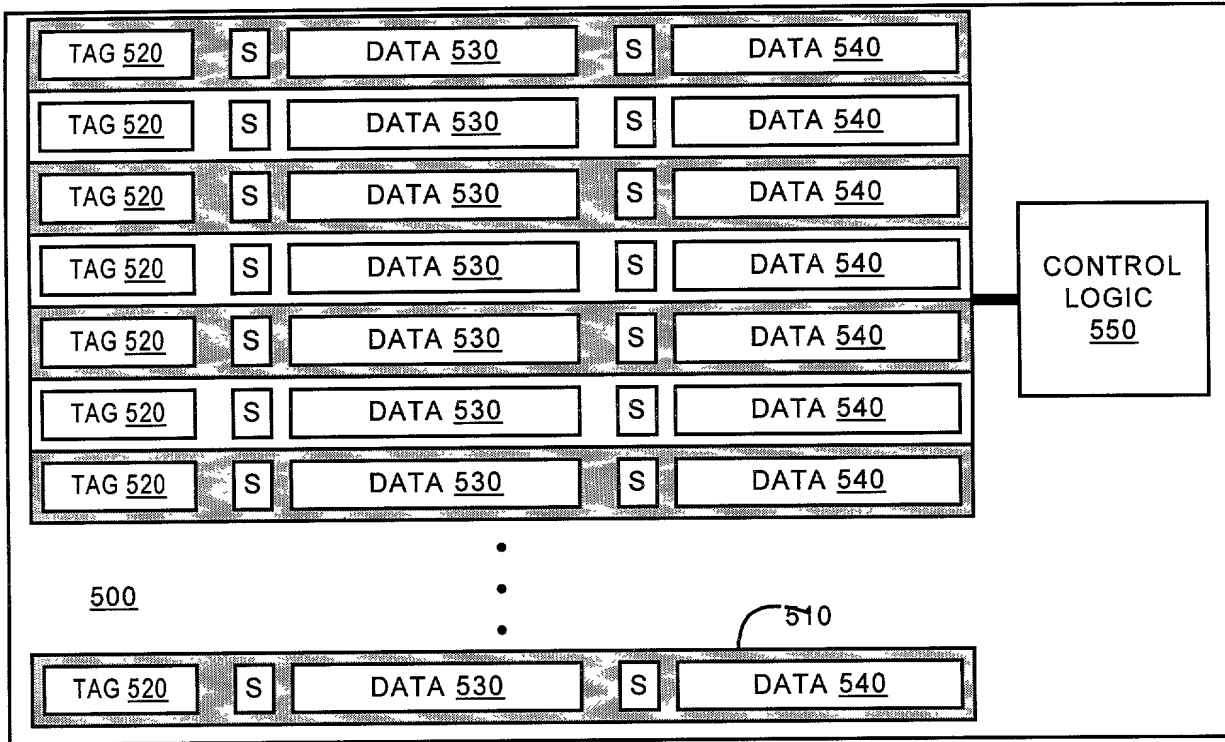




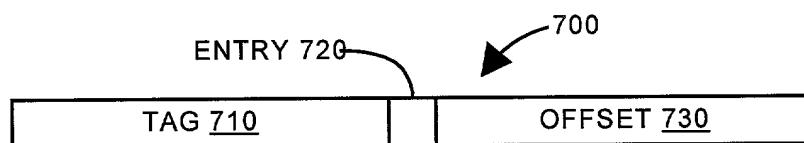
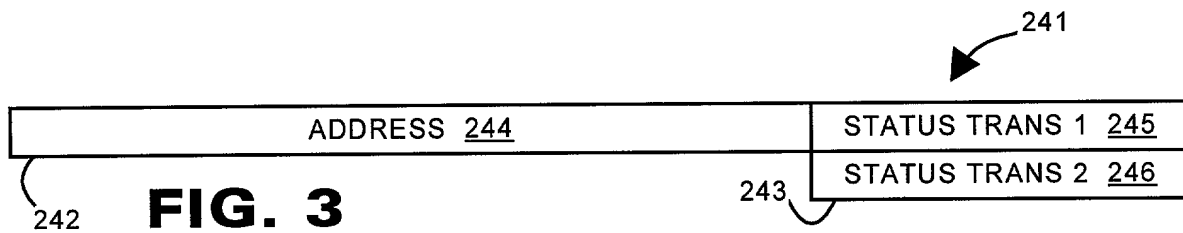
**FIG. 1**



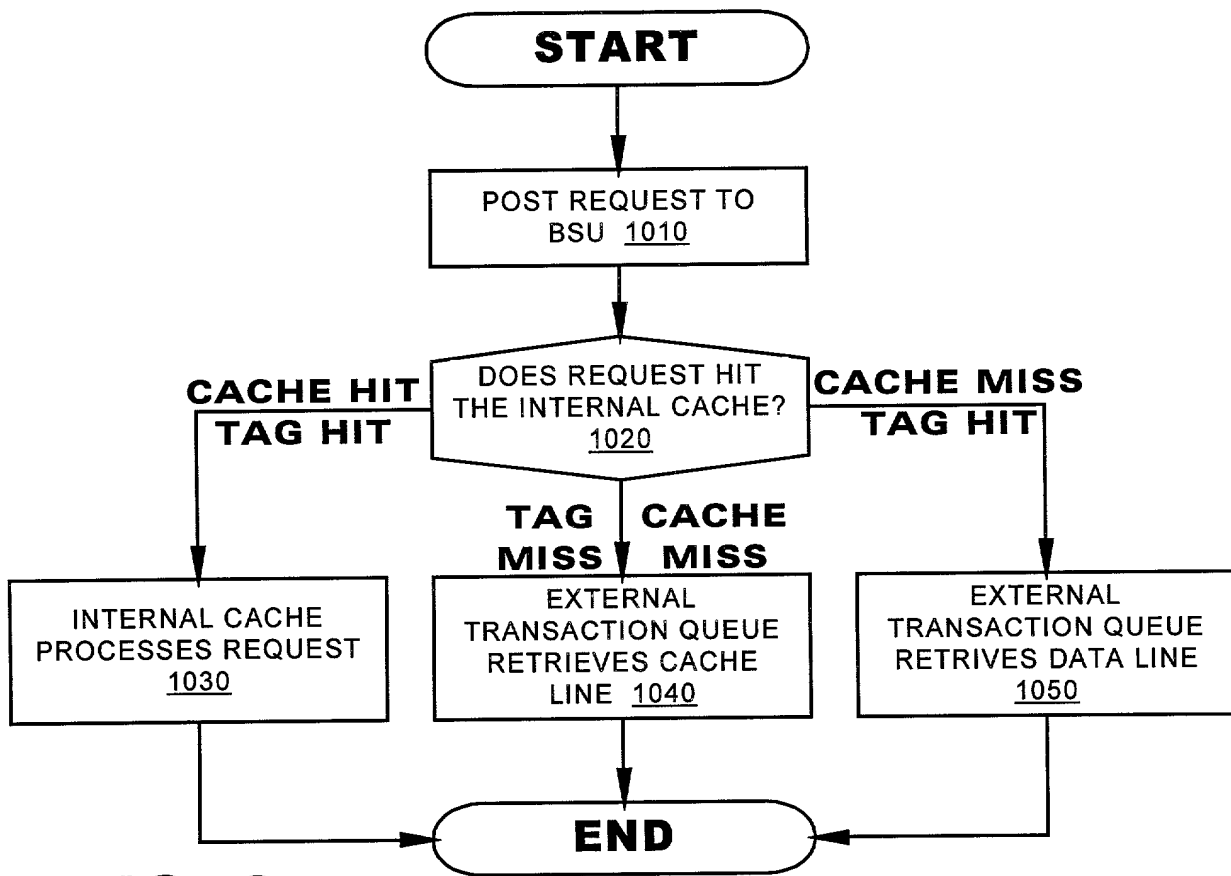
**FIG. 4**



**FIG. 2**

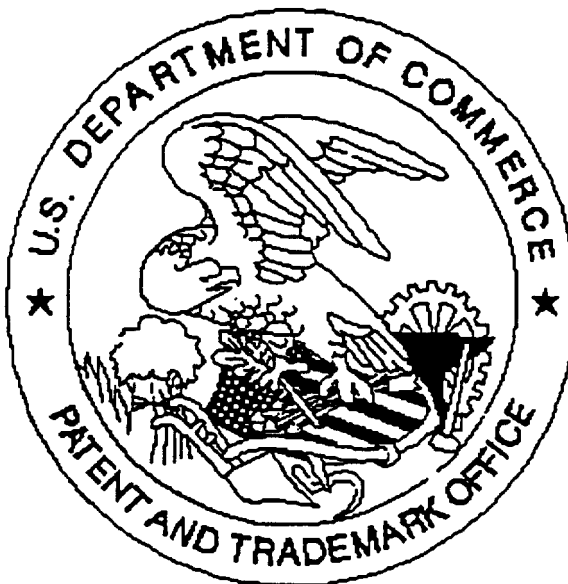


**FIG. 5**



**FIG. 6**

United States Patent & Trademark Office  
Office of Initial Patent Examination -- Scanning Division



Application deficiencies were found during scanning:

☐ Page(s) \_\_\_\_\_ of Declaration were not present  
for scanning. (Document title)

☐ Page(s) \_\_\_\_\_ of \_\_\_\_\_ were not present  
for scanning. (Document title)

☐ Scanned copy is best available.